FULL STACK DEVELOPMENT

TASK 2

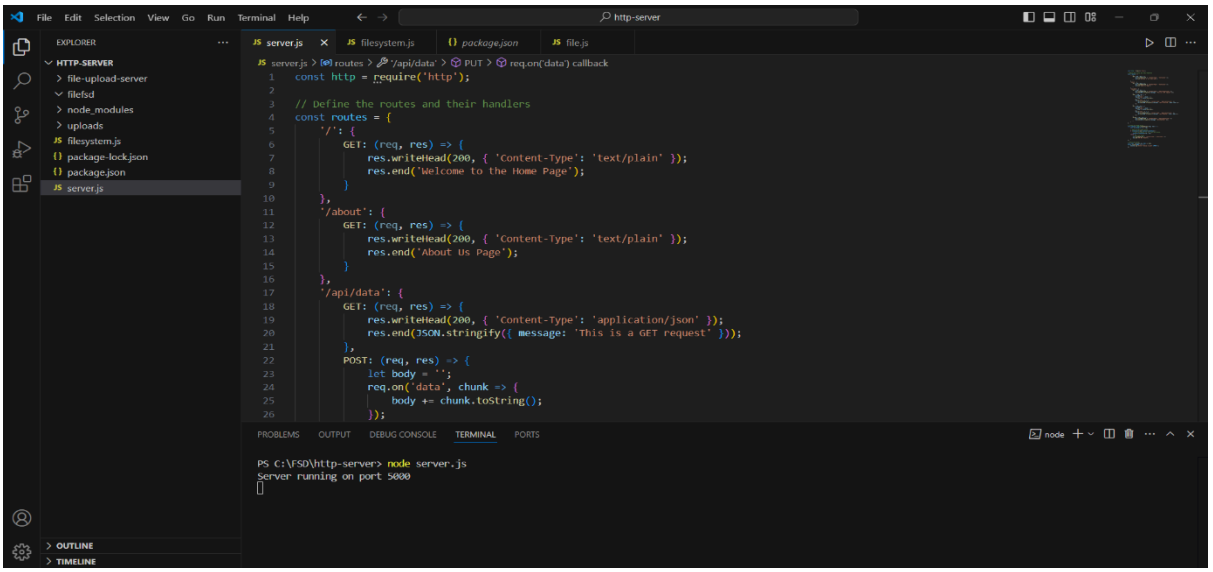
1.Build a basic HTTP server that can handle different routes and HTTP methods (GET, POST, PUT, DELETE).

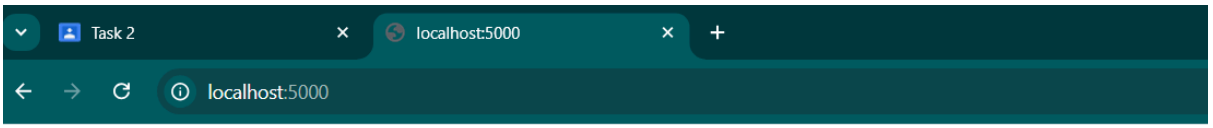
PROGRAM:

```
const http = require('http');

// Define the routes and their handlers

const routes = {

  '/': {

    GET: (req, res) => {

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.end('Welcome to the Home Page');

    }

  },

  '/about': {

    GET: (req, res) => {

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.end('About Us Page');

    }

  },

  '/api/data': {

    GET: (req, res) => {

      res.writeHead(200, { 'Content-Type': 'application/json' });

      res.end(JSON.stringify({ message: 'This is a GET request' }));

    },

    POST: (req, res) => {

      let body = '';

      req.on('data', chunk => {

        body += chunk.toString();

      });

      req.on('end', () => {

        res.writeHead(201, { 'Content-Type': 'application/json' });

        res.end(JSON.stringify({ message: 'Data received', data: body }));

      });

    },

    PUT: (req, res) => {
```

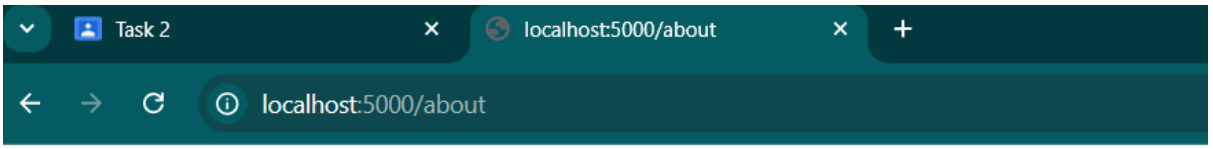
```
            let body = '';
            req.on('data', chunk => {
                body += chunk.toString();
            });
            req.on('end', () => {
                res.writeHead(200, { 'Content-Type': 'application/json' });
                res.end(JSON.stringify({ message: 'Data updated', data: body }));
            });
        },
        DELETE: (req, res) => {
            res.writeHead(200, { 'Content-Type': 'application/json' });
            res.end(JSON.stringify({ message: 'Data deleted' }));
        }
    }
};

// Create the HTTP server
const server = http.createServer((req, res) => {
    const { method, url } = req;

    // Check if the route exists
    if (routes[url] && routes[url][method]) {
        // Call the handler for the route and method
        routes[url][method](req, res);
    } else {
        // Route not found
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end('404 Not Found');
    }
});
// Start the server
const PORT = process.env.PORT || 5000;
server.listen(PORT, () => {
    console.log(`Server running on port ${PORT}`);
});
```

OUTPUT

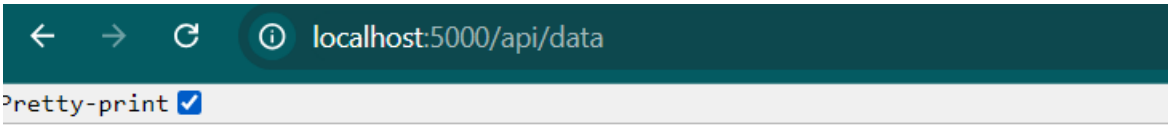




Welcome to the Home Page



About Us Page




```
{
  "message": "This is a GET request"
}
```

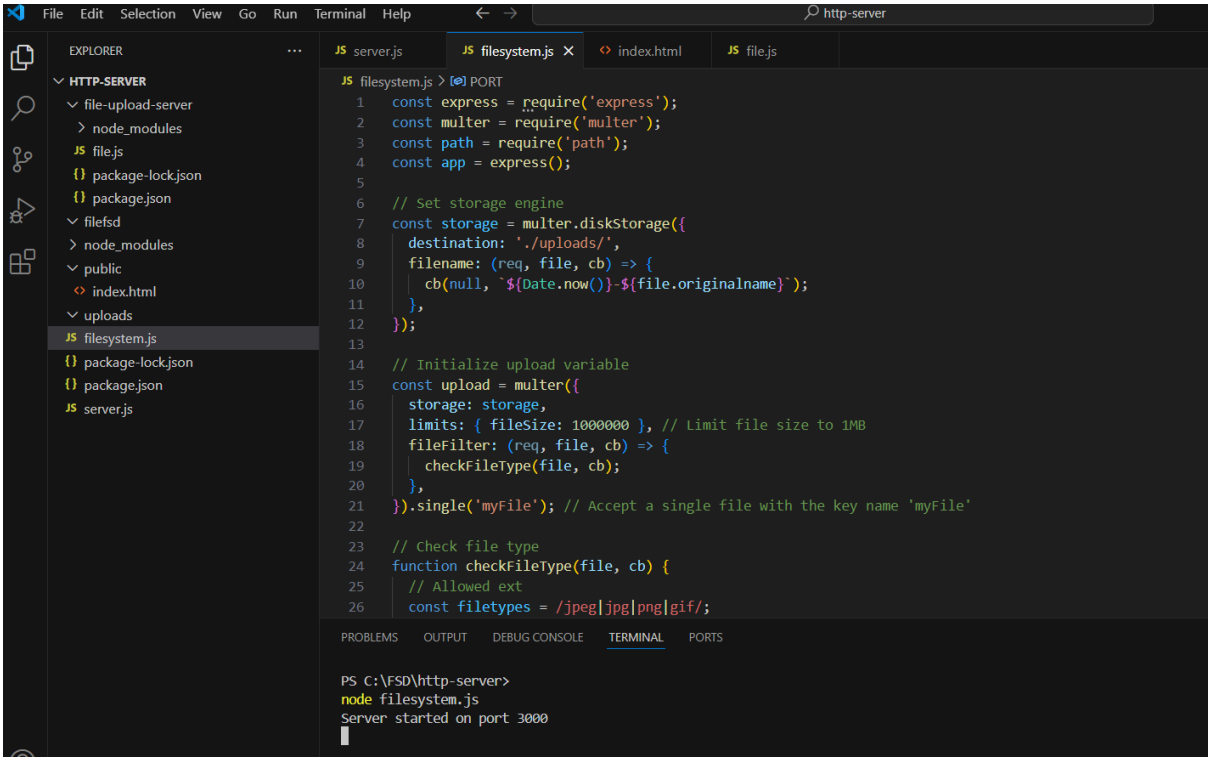
```
Command Prompt
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vishn>curl http://localhost:5000/api/data
{"message":"This is a GET request"}
C:\Users\vishn>curl -X POST http://localhost:5000/api/data -d "name=John&age=30"
{"message":"Data received","data":"name=John&age=30"}
C:\Users\vishn>curl -X PUT http://localhost:5000/api/data -d "name=Jane&age=25"
{"message":"Data updated","data":"name=Jane&age=25"}
C:\Users\vishn>curl -X DELETE http://localhost:5000/api/data
{"message":"Data deleted"}
C:\Users\vishn>curl -X POST http://localhost:5000/api/data -d "name=John&age=30"
{"message":"Data received","data":"name=John&age=30"}
C:\Users\vishn>
```

2. Create a server that allows users to upload files and save them to the server's filesystem.

PROGRAM

```
const express = require('express');

const multer = require('multer');

const path = require('path');

const app = express();

// Set storage engine

const storage = multer.diskStorage({

  destination: './uploads/',

  filename: (req, file, cb) => {

    cb(null, `${Date.now()}-${file.originalname}`);

  },

});

const upload = multer({

  storage: storage,

  limits: { fileSize: 1000000 }, // Limit file size to 1MB

  fileFilter: (req, file, cb) => {

    checkFileType(file, cb);

  },

}).single('myFile'); // Accept a single file with the key name 'myFile'

// Check file type

function checkFileType(file, cb) {
```

```
  const filetypes = /jpeg|jpg|png|gif/;
  const extname = filetypes.test(path.extname(file.originalname).toLowerCase());
  const mimetype = filetypes.test(file.mimetype);
  if (mimetype && extname) {
    return cb(null, true);
  } else {
    cb('Error: Images Only!');
  }
}
app.use(express.static('./public'));
app.get('/', (req, res) => res.send('File Upload Server'));
app.post('/upload', (req, res) => {
  upload(req, res, (err) => {
    if (err) {
      res.status(400).send(err);
    } else {
      if (req.file === undefined) {
        res.status(400).send('Error: No File Selected!');
      } else {
        res.send(`File uploaded: ${req.file.filename}`);
      }
    }
  });
});
const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log(`Server started on port ${PORT}`));
```


File Edit Selection View Go Run Terminal Help
http-server
EXPLORER
HTTP-SERVER
file-upload-server
node_modules
file.js
package-lock.json
package.json
filefsd
node_modules
public
index.html
uploads
filesystem.js
package-lock.json
package.json
server.js
server.js
filesystem.js
index.html
file.js
filesystem.js > PORT
const express = require('express');
const multer = require('multer');
const path = require('path');
const app = express();
// Set storage engine
const storage = multer.diskStorage({
destination: './uploads/',
filename: (req, file, cb) => {
cb(null, `${Date.now()}-${file.originalname}`);
},
});
// Initialize upload variable
const upload = multer({
storage: storage,
limits: { fileSize: 1000000 }, // Limit file size to 1MB
fileFilter: (req, file, cb) => {
checkFileType(file, cb);
},
}).single('myFile'); // Accept a single file with the key name 'myFile'
// Check file type
function checkFileType(file, cb) {
// Allowed ext
const filetypes = /jpeg|jpg|png|gif/;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\FSD\http-server>
node filesystem.js
Server started on port 3000

OUTPUT:


localhost:4000
File Upload Server